Cipher Breaking using Markov Chain Monte Carlo

Davit Karamyan , YSU Applied Mathematics and Programming

Abstract - In this paper, we talk about the Markov Chain Monte Carlo (MCMC) method to decrypt text encoded with a secret substitution cipher. Section I guides you through a Bayesian framework for deciphering a substitution cipher .In section II we will create the main algorithm for deciphering and in section III we will implement algorithm in Python. Also we will discuss how we can optimize and develop solution using linear algebra techniques and natural language processing methods in section IV.

Keywords - MCMC, python, cipher breaking , inference

INTRODUCTION

A substitution cipher is a method of encryption by which units of plaintext — the original message — are replaced with *cipher-text* —the encrypted message — according to a ciphering function. The "units" may be single symbols, pairs of symbols, triplets of symbols, mixtures of the above, and so forth. The decoder deciphers the text by inverting the cipher.

For example, consider a simple substitution cipher that operates on single symbols.

The ciphering function f is a one-to-one mapping between two finite and equal size alphabets $\mathcal{A}$ and $\mathcal{B}$. The plaintext is a string of symbols, which can be represented as a length-n vector $x = (x_1,\ldots, x_n) \in \mathcal{A}^n$. Similarly, the *cipher-text* can be represented as vector $y = (y_1,\ldots, y_n) \in \mathcal{B}^n$ such that

$$y_k = f(x_k);\ k = 1,2,\ldots,n$$

Without loss of generality, we assume identical alphabets, $\mathcal{A} = \mathcal{B}$, so that a ciphering function f is merely a permutation of symbols in $\mathcal{A}$.

Throughout this project, we restrict our attention to the alphabet $\mathcal{A} = \mathcal{E} \cup \{␣,.\}$ where $\mathcal{E} = \{a, b,\ldots,z\}$ set of lower-case English letters.

Deciphering a *cipher-text* is straightforward if the ciphering function f is known.

Specially, *cipher-text* $y = (y_1,\ldots, y_n)$ is decoded as $x_k = f^{-1}(y_k)$; $k = 1,2,\ldots,n$ where the deciphering or decoding function f^{-1} is the functional inverse of f. Note that this inverse exists because f is one-to-one. However, when f is a secret (i.e., unknown), decoding a *cipher-text* is more involved and is more naturally framed as a problem of inference. In this project, you will develop an efficient algorithm for decoding such secret *cipher-texts*.

I.BAYESIAN FRAMEWORK

Suppose a process is generating independent and identically distributed events En, but the probability distribution is unknown. Let the event space Ω represent the current state of belief for this process. Each model is represented by event Mm. The conditional probabilities $P(En \mid Mm)$ are specified to define the models. $P(Mm)$ is the degree of belief in Mm. Before the first inference step, $\{P(Mm)\}$ is a set of initial prior probabilities.

Suppose that the process is observed to generate $E \in \{En\}$. For each $M \in \{Mm\}$, the prior $P(M)$ is updated to the posterior $P(M \mid E)$). From Bayes' theorem:

$$P(M \mid E) = \frac{P(E \mid M)}{\sum_m P(E \mid M_m)P(M_m)} \cdot P(M)$$

UNDERSTANDING MONTE CARLO METHODS

Monte Carlo methods vary, but tend to follow a particular pattern:

1)Define a domain of possible inputs

2) Generate inputs randomly from a probability distribution over the domain

3)Perform a deterministic computation on the inputs

4)Aggregate the results

Let's consider example how its works.

Finding areas: Now we want to estimate the area of some shape given equations of this shape. If we try to find area using calculus techniques it will be very infeasible.

Inputs: Equations of shape

Output: The area of shape

Without loss of generality, let's consider that this shape is a circle(C) , with center(C) = (1/2,1/2) and with radius(C) = 1/2 and therefore with equation $(x-0.5)^2 + (y-0.5)^2 = 0.25$ .Note that the shape can be any closed figure on plain.

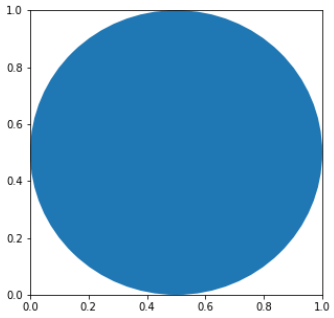


fig.1

It is pretty easy to calculate the area of circle $S(C)=\pi R^2 = 0.785$.

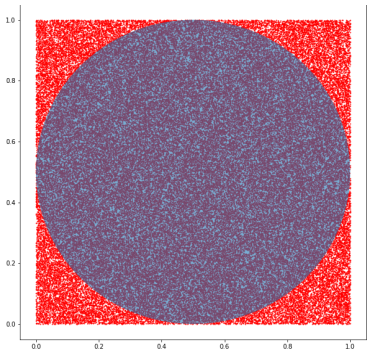
Let's try to solve this problem using MC. Consider the square A=[0:1][0:1]. So we can see from fig.1 that $C \subset A$. And let's assume that there is a uniform distribution over square A. We can easily generate samples from this distribution. The point (a, b) will be in circle if a and b satisfy this inequality: $(a-0.5)^2 + (b-0.5)^2 <= 0.25$ .If the number of samples(N) is a pretty huge, then using this formula we can estimate S (C) :

$$S(C) = \frac{\text{number of points in circle}}{N}$$

Here is a code for generating points.

```
>>>from matplotlib import patches
import matplotlib.pyplot as plt
import numpy as np

plt.figure(figsize=(5,5))
plt.clf()
inCircle = []
inSquere = []
for i in range(1,100000):
    a = random.uniform(0,1)
    b = random.uniform(0,1)
    inSquere.append([a,b])
    if (a-0.5)**2 + (b-0.5)**2 <= 0.25:
        inCircle.append([a,b])
plt.scatter(np.array(inSquere)[:,0],
            np.array(inSquere)[:,1],
            s=1, color='red')
e=patches.Circle((0.5,0.5), 0.5)
e.set_alpha(.6)
plt.axes().add_artist(e)
plt.savefig('Circle')
S = len(inCircle)/len(inSquere)
print("S = " ,S)
>>> S = 0.7856178561
```



Let's extend this example into general algorithm which is known as *Rejection sampling* algorithm.

REJECTION SAMPLING

rejection sampling is a basic technique used to generate observations from a distribution. Let's do some notation.

$$p(x) = \frac{\tilde{p}(x)}{Z_p} - \text{target distrubution}$$

$q(x) = \frac{\tilde{q}(x)}{Z_q}$ – proposal distrubution

Our main goal is to generate samples from target distribution p(x).But we cannot generate samples from p(x) directly, because it's not easy. Alternatively , we take another distribution q(x) from which we can easily generate samples. The only restriction on q(x) is this:

$$\exists c > 0 \forall x \in X \; c\tilde{q}(x) > \tilde{p}(x)$$

Rejection Algorithm

. Generate x from $q(x)$

. Generate u from $U[0,c\tilde{q}(x)]$

. If $u <= \tilde{p}(x)$: keep x else: do step 1

Taking back to the example for finding areas , we can not easily work with circle (i.e p(x)).
So we included circle into square(i.e cq(x)).
Then we generate samples from q(x) which is an uniform distribution over square A. if the generated point in circle (i.e u <= p(x)) then we keep x .

MARKOV CHAIN

Definition: Roughly speaking, a process satisfies the Markov property if one can make predictions for the future of the process based solely on its present state just as well as one could knowing the process's full history, hence independently from such history; i.e., conditional on the present state of the system, its future and past states are independent. More mathematically:

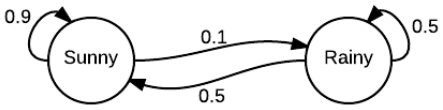
$$P(x_n | x_{n-1}, \ldots, x_1) = P(x_n | x_{n-1})$$

A SIMPLE WEATHER MODEL

The probabilities of weather conditions (modeled as either rainy or sunny), given the weather on the preceding day, can be represented by a transition matrix:

$$V = \begin{bmatrix} \alpha = 0.9 & \beta = 0.1 \\ 1-\alpha = 0.5 & 1-\beta = 0.5 \end{bmatrix}$$

The matrix V represents the weather model in which a sunny day is 90% likely to be followed by another sunny day, and a rainy day is 50% likely to be followed by another rainy day. The columns can be labelled "sunny" and "rainy", and the rows can be labelled in the same order. $(V)_{ij}$ is the probability that, if a given day is of type i, it will be followed by a day of type j. Notice that the rows of P sum to 1.



Predicting the weather:The weather on day 1 is known to be sunny. This is represented by a vector in which the "sunny" entry is 100%, and the "rainy" entry is 0%:

$$\mathbf{x}^{(0)} = [1 \quad 0]$$

The weather on day 2 can be predicted by:

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)}V = [1 \quad 0] \begin{bmatrix} 0.9 & 0.1 \\ 0.5 & 0.5 \end{bmatrix} = [0.9 \quad 0.1]$$

Thus, there is a 90% chance that day 2 will also be sunny.
General rules for day n are:

$$\mathbf{x}^{(n)} = \mathbf{x}^{(n-1)}V$$
$$\mathbf{x}^{(n)} = \mathbf{x}^{(0)}V^n$$

Here we can not discuss about steady state, only the definition will be covered.
Steady state vector represents the probabilities of sunny and rainy weather on all days, and is independent of the initial weather.
The steady state vector is defined as:

$$\mathbf{q} = \lim_{n \to \infty} \mathbf{x}^{(n)}$$

Y S U

METROPOLIS–HASTINGS ALGORITHM

In statistics and in statistical physics, the **Metropolis–Hastings algorithm** is a Markov chain Monte Carlo (MCMC) method for obtaining a sequence of random samples from a probability distribution for which direct sampling is difficult. This sequence can be used to approximate the distribution (e.g., to generate a histogram), or to compute an integral (such

given the target distribution - $\frac{\tilde{p}(x)}{Z_p}$

given the transition matrix $V(|)$ from X

Metropolis-Hastings algorithm

step 1) start with arbitrary x_0 from X

step 2) At time n $x = x_n$

step 3) Generate $\acute{x}$ from $V(|x)$

step 4)$\alpha(x -> \acute{x}) = min\{1, \frac{p(\acute{x})V(x|\acute{x})}{p(x)V(\acute{x}|x)}\}$

step 5) Generate $u \in \{A, R\}$ from Bernulii $p(u = A) = \alpha(x -> \acute{x})$

step 6) If u == A(Accept) : $X_{n+1} = \acute{x}$ else: $X_{n+1} = x$

step 7) got to step 2)

II. DESIGN ALGORITHM FOR CIPHER BREAKING

In this part we address the problem of inferring the plaintext x from the observed *cipher-text* y in a Bayesian framework. For this purpose, we model the ciphering function f as random and drawn from the uniform distribution over the set of permutations of the symbols in alphabet A. Assume that characters in English text can be approximately modeled as a simple Markov chain, so that the probability of a symbol in some text depends only on the symbol that precedes it in the text.
Enumerating the symbols in the alphabet A from 1 to m = |A| for convenience, the probability of transitioning from a symbol indexed with i to a symbol indexed with j is given by:

$$p(x_k = i \,|\, x_{k-1} = j) = M_{ij}$$

i.e., the ith row, jth column of the matrix $\mathcal{M} = [\mathcal{M}_{i,j}]$ has the probability of transition from symbol j at time k—1 to symbol i at time k.
Moreover, assume that probability of the ith symbol in A is given by:

$$p(x_k = i) = P_i$$

Finally, assume that M and P = [Pi] are known (given).

To see implementation go to Github\